(19)

(11) Publication number: **09185528 A**

PATENT ABSTRACTS OF JAPAN

(21) Application number: **08000462**

(51) Intl. Cl.: **G06F 11/28** G06F 9/46

(22) Application date: **08.01.96**

(30) Priority:

(43) Date of application publication: **15.07.97**

(84) Designated contracting states:

(71) Applicant: **CANON INC**

(72) Inventor: **SASAKI SEIJI**
HIRAHARA ATSUSHI
WATANABE HIROYASU

(74) Representative:

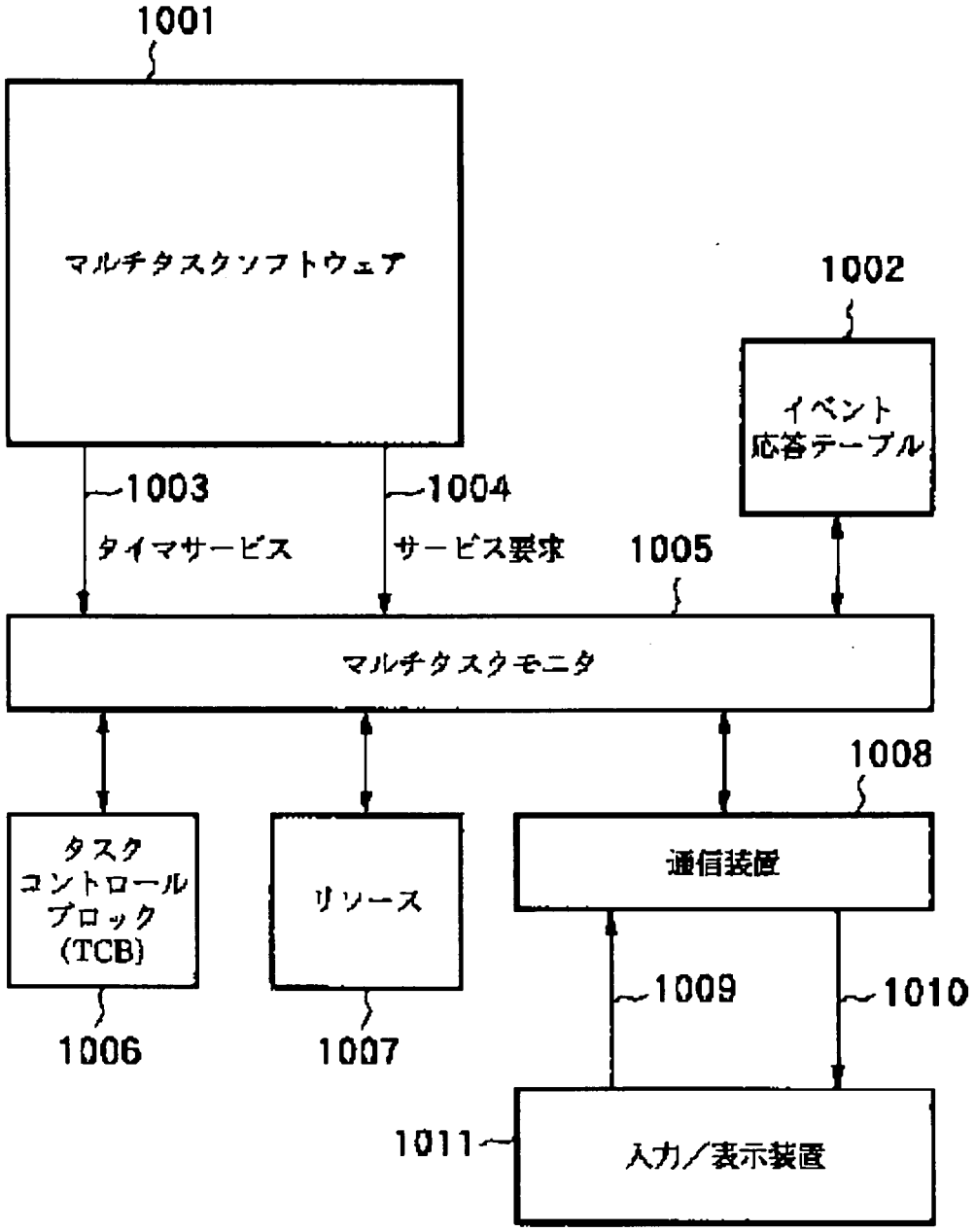
(54) METHOD AND DEVICE FOR INSPECTING SOFTWARE

(57) Abstract:

PROBLEM TO BE SOLVED: To minutely inspect the operation of multi-task software for detecting the error of inter-task interface.

SOLUTION: Multi-task software 1001 operating on hardware loading a low speed communication equipment is inspected by operating multi-task software 1001 on a real time basis

and automatically detecting violence against interface specification between tasks, which is previously set, and incorporating an inspection method for inspecting multi-task software 1001 into a multi-task monitor 1005.


1001
マルチタスクソフトウェア
1002
イベント
応答テーブル
1003
タイマサービス
1004
サービス要求
1005
マルチタスクモニタ
1008
タスク
コントロール
ブロック
(TCB)
リソース
通信装置
1009
1010
1006
1007
1011
入力／表示装置

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（A）　(11)特許出願公開番号

特開平9－185528

(43)公開日　平成９年(1997)７月15日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０６Ｆ　11/28	３４０	7313－5Ｂ	Ｇ０６Ｆ　11/28	３４０Ａ	
9/46	３３０		9/46	３３０Ｃ	

審査請求　未請求　請求項の数６　ＯＬ　（全 18 頁）

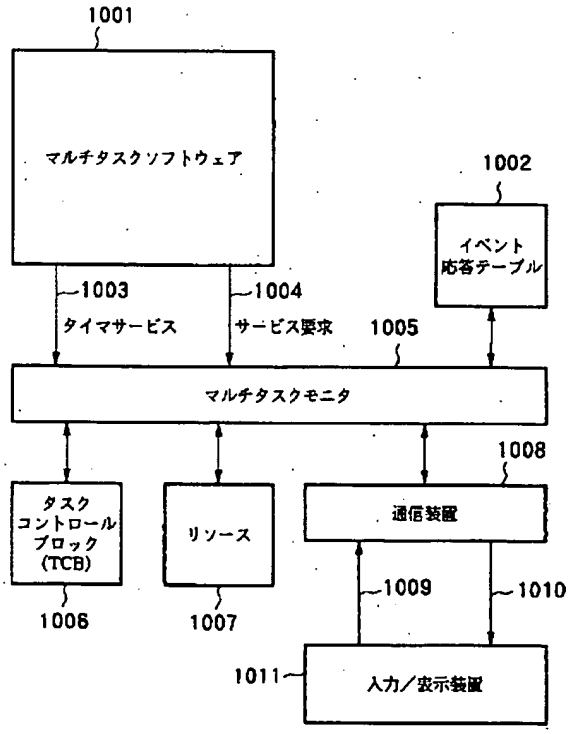
(21)出願番号　特願平8－462

(22)出願日　平成８年(1996)１月８日

(71)出願人　000001007
キヤノン株式会社
東京都大田区下丸子３丁目30番２号
(72)発明者　佐々木　誠司
東京都大田区下丸子３丁目30番２号　キヤノン株式会社内
(72)発明者　平原　厚志
東京都大田区下丸子３丁目30番２号　キヤノン株式会社内
(72)発明者　渡辺　浩康
東京都大田区下丸子３丁目30番２号　キヤノン株式会社内
(74)代理人　弁理士　大塚　康徳　（外１名）

(54)【発明の名称】　ソフトウェア検査方法およびその装置

(57)【要約】
【課題】　タスク間インタフェイスのエラーを検出するには、マルチタスクソフトウェアの動作を詳細に調べる必要がある。
【解決手段】　マルチタスクソフトウェアを実時間で動作させ、予め設定されたタスク間のインタフェイス仕様に対する違反を自動的に検出することにより、マルチタスクソフトウェアの検査を行う検査方法をマルチタスクモニタへ組込むことにより、低速な通信装置を搭載するハードウェア上で動作するマルチタスクソフトウェアの検査を行う。



【特許請求の範囲】
【請求項1】　マルチタスクソフトウェアを実時間で動作させ、予め設定されたタスク間のインタフェイス仕様に対する違反を検出することにより、前記マルチタスクソフトウェアの検査を行うことを特徴とするソフトウェア検査方法。
【請求項2】　さらに、各タスクおよびタスク間で共通にアクセスされるリソースに対するイベントと、そのイベントに対するレスポンスのペア集合により、タスク間のインタフェイス仕様を表現することを特徴とする請求項1に記載されたソフトウェア検査方法。
【請求項3】　前記リソースはキュー，セマフォ，イベントフラグを含むことを特徴とする請求項2に記載されたソフトウェア検査方法。
【請求項4】　請求項1に記載されたソフトウェア検査方法をマルチタスクモニタへ組込むことにより、低速な通信装置を搭載するハードウェア上で動作するマルチタスクソフトウェアの検査を行うことを特徴とするソフトウェア検査装置。
【請求項5】　前記マルチタスクモニタは、そのテーブルに外部から設定された前記マルチタスクソフトウェアのタスク間インタフェイス仕様にかかわる動作を監視し、前記通信装置を用いてインタフェイスプロトコル違反を外部へ通知することを特徴とする請求項4に記載されたソフトウェア検査装置。
【請求項6】　前記マルチタスクモニタは、前記タスク間インタフェイス仕様にかかわるイベントの発生を検出した場合は対応するレスポンスを励起状態にし、所定の許容時間内にレスポンスが実行されない場合は外部へエラーを通知することを特徴とする請求項5に記載されたソフトウェア検査装置。
【発明の詳細な説明】
【0001】
【発明の属する技術分野】本発明はソフトウェア検査方法およびその装置に関し、例えば、ハードウェアに組込まれるとソフトウェアなど実時間動作を行うソフトウェア検査方法およびその装置に関するものである。
【0002】
【従来の技術】ソフトウェア検査は、通常、プログラムデバッガを用いて行われる。プログラムデバッガは、プログラムソースの任意行でプログラムの実行停止と再開ができるとともに、実行停止時にはプログラム変数やフラグの状態表示および変更を行うことができる。従って、ソフトウェアの開発者は、プログラムデバッガを対話的に使用し、ソフトウェアのバグを探すとともに、そのバグを修正する作業を繰返して、ソフトウェアを完成させる。
【0003】より大規模な実時間システムでは、マルチタスク機構に基づいたタスク分割プログラミング手法が一般に利用されているが、マルチタスク処理に特有な、タスク間の制御の移り変わり，メッセージの送受信，同期化機能を、外部記憶装置に蓄積するようなタスク間イベントのトレーサが実用化されている。従って、タスク分割プログラミングにおいては、タスク間イベントトレーサを用いてタスク間のインタフェイスの検証を行うとともに、プログラムデバッガを用いて個別のタスクを検証する方法が、現時点で最も効率的なソフトウェアの検証方法であると考えられる。
【0004】
【発明が解決しようとする課題】しかし、上述した技術においては、次のような問題点がある。
【0005】ハードウェア制御を目的にするソフトウェアにおいては、プログラムの実行を止めると制御対象のハードウェアが破損する場合があるので、必ずしも、プログラムの実行を停止させることができない。このため、このような目的のソフトウェアにおいては、プログラムデバッガを効率的に利用することができない。
【0006】また、マルチタスクソフトウェアの開発においても、タスク間イベントトレーサは、イベント系列を報告/表示するのみであり、そのトレース結果から、マルチタスクソフトウェアの動作シーケンスの正当性に関する判断は人手に頼っている。通常、タスク間イベントは、数msから数10ms単位で発生するため、ソフトウェアの動作が短時間であっても、膨大な量のトレース結果が生成され、通常、その正当性の判断は不可能である。
【0007】従って、マルチタスクソフトウェアの不具合現象に対して、その現象の発生を含むトレース結果を丹念に調べ、不具合の直接の原因を探るという方法が最も有効であると思われるが、従来のソフトウェア検査装置がソフトウェアの不具合を検出する能力はさほど高くない。
【0008】さらに、トレーサは、膨大な量のトレースデータを外部記憶装置に転送するため、マルチタスクソフトウェアが搭載されるハードウェアは、高速通信機能を備えていなければならず、そうでなければ、ハードウェアに搭載するマルチタスクソフトウェアの開発において、トレーサを利用することができない。
【0009】本発明は、上述の問題を解決するためのものであり、マルチタスクソフトウェアの動作を詳細に調べることなく、タスク間インタフェイスのエラーを検出することができるソフトウェア検査方法およびその装置を提供することを目的とする。
【0010】
【課題を解決するための手段】本発明は、前記の目的を達成する一手段として、以下の構成を備える。
【0011】本発明にかかるソフトウェア検査方法は、マルチタスクソフトウェアを実時間で動作させ、予め設定されたタスク間のインタフェイス仕様に対する違反を検出することにより、前記マルチタスクソフトウェアの検査を行うことを特徴とする。

【００１２】また、本発明にかかるソフトウェア検査装置は、マルチタスクソフトウェアを実時間で動作させ、予め設定されたタスク間のインタフェイス仕様に対する違反を検出することにより、前記マルチタスクソフトウェアの検査を行うソフトウェア検査方法をマルチタスクモニタへ組込むことにより、低速な通信装置を搭載するハードウェア上で動作するマルチタスクソフトウェアの検査を行うことを特徴とする。

【００１３】

【発明の実施の形態】以下、本発明にかかる一実施形態のソフトウェア検査装置を図面を参照して詳細に説明する。

【００１４】［機能構成］図1は本発明にかかる一実施形態のソフトウェア検査装置の構成例を示す機能ブロック図である。

【００１５】同図において、イベント応答テーブル（以下、単に「テーブル」と呼ぶ場合がる）1002は、検査対象のマルチタスクソフトウェア1001のタスク間インタフェイス仕様を表現したイベント応答ペアからなる。マルチタスクモニタ1005は、マルチタスク動作を実現し、テーブル1002のタスク間インタフェイス仕様に基づく、マルチタスクソフトウェア1001の検証機能を備えている。そして、マルチタスクモニタ1005は、タスクの状態を保持するタスクコントロールブロック(TCB)1006と、タスク間通信およびタスク間同期機能を実現するためのリソース1007を管理している。

【００１６】マルチタスクソフトウェア1001は、マルチタスクモニタ1005に対して、サービス要求1004を発行することで、タスク状態の変更、および、リソースへのアクセスを行うことができる。また、マルチタスクソフトウェア1001は、ハードウェアタイマからの定期的なタイマ割込に応答して、タイマサービス1003を呼出し、マルチタスクモニタ1005にシステムクロックを提供する。

【００１７】マルチタスクモニタ1005は、マルチタスクソフトウェア1001の実行に必要なタスクおよびリソースを生成した時点で、通信装置1008を介して入力/表示装置1011から、マルチタスクソフトウェア1001のタスク間インタフェイス仕様を表現するイベント-応答リストを読込み、テーブル1002に保存する。

【００１８】マルチタスクモニタ1005は、サービス要求1004を通して発行されるリソースへの送信/受信/受信要求などのイベントが、テーブル1002のイベント部分と一致する場合、対応する応答（アクション）が所定の許容時間内で実行されることを監視する。発生したイベントに対する応答が許容時間内に実行されない場合、マルチタスクモニタ1005は、通信装置1008を介して入力/表示装置1011にエラーを報告する。

【００１９】［マルチタスクソフトウェア］図2は典型的なマルチタスクソフトウェアを示す図である。

【００２０】同図において、割込ハンドラ2001はハードウェアからの割込2041を処理し、タイマ割込ハンドラ2002はハードウェアタイマからの割込2042を処理する。また、2011～2013はそれぞれ固有の処理を実現するタスクである。

【００２１】タスク間通信を実現するキュー2021は割込ハンドラ2001からタスク2011へ情報を渡し、同様に、タスク間通信を実現するキュー2022はタスク2012からタスク2013へ情報を渡すために使用される。

【００２２】タスク間同期機能を実現するイベントフラグ2031のビット0は、タスク2011によりセットされ、タスク2012とタスク2011の同期に利用される。イベントフラブ2031のビット1は、タスク2013によりセットされ、タスク2011とタスク2013の同期に利用される。

【００２３】図3は図2に示すマルチタスクソフトウェアの時系列的な動作例を示すシーケンス図である。なお、このシーケンス図においては、タスクの優先度が2011>;2012>;2013の順であり、割込ハンドラ2001の優先度が各タスクよりも高い場合を想定している。

【００２４】タイミングt0においてタスク2011,2012,2013は中断状態にあり、タイミングt1において割込み2041が発生して割込ハンドラ2001が起動される。割込ハンドラ2001は、タイミングt2でキュー2021への送信イベント2051を発行する。またこのとき、タスク2011は、タイミングt0で要求したキュー受信要求2052が満たされ、中断状態から実行待ち状態に遷移する。

【００２５】タイミングt3において、割込ハンドラ2001が終了するため、唯一、実行可能な状態（実行状態または実行待ち状態）のタスク2011に制御が切替わる。

【００２６】タイミングt4において、タスク2011はイベントフラグ2031への送信イベント2053を発行し、タスク2012はイベントフラグ2031から受信イベント2054を受取って実行待ち状態に遷移する。

【００２７】タイミングt5において、タスク2011はイベントフラグ2031への受信要求2058を発行して中断状態に遷移する。このとき、タスク2012だけが実行可能状態になるので、タイミングt6において、タスク2012はキュー2022への送信イベント2055を発行し、タスク2013はキュー2022から受信イベント2056を受取り、タスク2013は中断状態から実行待ち状態に遷移する。

【００２８】タイミングt7において、タスク2012はイベントフラグ2031への受信要求2054を発行し、中断状態に遷移する。このとき、タスク2013が、唯一、実行可能な状態になるので、タスク2013は実行状態に遷移する。

【００２９】タイミングt8において、タスク2013は、イベントフラグ2031への送信イベント2057を発行し、実行待ち状態に遷移する。また、より優先度の高いタスク2011は、イベントフラグ2031から受信イベント2058を受取り、実行状態に遷移する。

【００３０】タイミングt9において、タスク2011はキュー2021への受信要求2052を発行して中断状態になり、タ

スク2013が実行状態になる。そして、タイミングt10において、タスク2013はキュー2022への受信要求2056を発行して中断状態になり、この時点で、すべてのタスクはタイミングt0と同様の初期状態に戻る。

【００３１】図4はタスク間インタフェイスプロトコルをイベント-応答表現で表す図で、図2に示すマルチタスクシステムが図3のシーケンス図に従って動作するために守らなければならないプロトコルである。

【００３２】同図に示すように、キュー2021は、送信イベント2051が発生したならば、受信イベント2052により応答しなければならない。また、キュー2022は、送信イベント2055が発生したならば、受信イベント2056により応答しなければならない。また、イベントフラグ2031は、送信イベント2053に対して受信イベント2054で、送信イベント2057に対して受信イベント2058で、それぞれ応答しなければならない。

【００３３】タスク2011においては、受信イベント2052に対して、送信イベント2053とそれに続く受信要求イベント2058が引き起こされることが示され、受信イベント2058に対しては受信要求イベント2052が引き起されることが示されている。

【００３４】また、タスク2012においては、受信イベント2054に対して、送信イベント2055とそれに続く受信要求イベント2054が引き起こされることが示され、同様に、タスク2013においては、受信イベント2056に対して、送信イベント2057とそれに続く受信要求イベント2056が引き起こされることが示されている。

【００３５】上記のプロトコルがすべて満足されている限り、図2に示すマルチタスクソフトウェアは、タスクスケジュール時に実行可能なタスクの内、どのタスクが先に実行されるかの違いを除いて、図3に示すシーケンス図に従って動作することが保証される。

【００３６】以下では、リソースに対するより複雑なタスク間インタフェイスの場合のイベント-応答表現を検証する。

【００３７】［複雑なタスク間インタフェイス（タイプ1）］図5は複雑なタスク間インタフェイスの一例を示す図で、一つのキューに対して複数のタスクが送信し、さらに、複数のタスクが受信するようなタスク間インタフェイスである。

【００３８】同図において、キュー5001はタスク間でデータの受け渡しを行うものであり、タスク5011と5012はキュー5001にデータを送信するタスクであり、タスク5021と5022はキュー5001からデータを受信するタスクである。

【００３９】タスク5021または5022は、キュー5001に対して受信要求を発行すると、キュー5001内にデータがあれば、そのデータがキュー5001から送られてくる。また、キュー5001が空であれば、キュー5001がデータを受信するまで、その受信要求はブロックされる。

【００４０】タスク5011または5012が、キュー5001へデータを送信した場合、それらのタスクはブロックされない。

【００４１】図6は図5に示すタスク間インタフェイスの動作例を示すシーケンス図で、送受信の順序関係により場合分けした例を示している。

【００４２】ケースAは、受信タスクの一つ（図では5021）により先に受信要求が発行された状態で、送信タスクの一つ（図では5011）がキュー5001に対して送信イベントを発行した場合を示している。この場合、キュー5001へ受信要求を発行しブロックされていたタスク5021は、タスク5011が送信イベントを発行した時点で解放され、実行可能な状態になる。

【００４３】ケースBは、先に送信タスクの一つ（図では5011）がキュー5001に対して送信イベントを発行し、その後、受信タスクの一つ（図では5021）がキュー5001に受信要求を発行した場合を示している。この場合、受信タスク5021の受信要求は直ちに受入れられ、受信タスク5021は、送信タスク5011が送信したデータを受信する。

【００４４】ケースCは、二つの送信タスクがキュー5001に対して送信イベントを発行し、その後、受信タスクの一つ（図では5021）がキュー5001に対して受信イベントを続けて二つ発行した場合を示している。この場合、受信タスク5021は、ブロックされることなく、送信されたデータを先着順に（先入り先出し方式で）受取る。キュー5001へ続けて送信イベントを発行する場合、同数の受信イベントが引き起こされるので、キュー5001への送信イベントは累積効果をもつ。

【００４５】ケースDは、二つの受信タスクがキュー5001に対して受信要求を発行した後、送信タスクの一つ（図では5011）がキュー5001に対して送信イベントを発行した場合を示している。この場合、ブロックされていた二つの受信タスクの内、優先順位の高いタスク（図ではタスク5022）が送信データを受信し、実行可能な状態になる。もう一方の受信タスク5021は中断状態のままである。

【００４６】図7は図5に示すタスク間インフェイスを備えたマルチタスクソフトウェアが図6に示すシーケンス図に従って動作するために必要なイベント-応答プロトコルを示す図である。

【００４７】同図(a)は、忠実にイベント-応答プロトコルを表現したものであり、イベントおよび応答がOR条件になっている。

【００４８】同図(b)は、同図(a)に示すプロトコルを二分割したもので、キュー5001に対する送信イベントは累積効果があるため、二つのプロトコルがともに満足されることは、同図(a)に示すプロトコルが満足されることと等価である。

【００４９】同図(c)は、同図(a)に示すプロトコルにお

いて、どの送信タスクが送信したかを問わないプロトコルである。キュー5001に送信するタスクが5011と5012に限られていることが確実であれば、同図(c)のプロトコルは同図(a)のプロトコルと等価である。

【００５０】同図(d)は、同図(b)に示すプロトコルにおいて、どの受信タスクが受信するかを問わないプロトコルである。キュー5001から受信するタスクが5021と5022に限られていることが確実であれば、同図(d)のプロトコルの両立は同図(a)のプロトコルと等価である。

【００５１】同図(e)は、同図(c)に示すプロトコルにおいて、どの受信タスクが受信するかを問わない（または、同図(d)に示すプロトコルにおいて、どの送信タスクかを問わない）プロトコルである。キュー5001に送信するタスクが5011と5012に限られ、キュー5001から受信するタスクが5021と5022に限られる状況では、同図(e)に示すプロトコルと同図(a)に示すプロトコルとは等価である。

【００５２】［複雑なタスク間インタフェイス（タイプ2）］図8は複雑なタスク間インタフェイスの第二例を示す図で、タスク間同期機能を果たす一つのイベントフラグ8001と、複数の送信タスク8011と8012と、同一ビット位置を複数の受信タスク8021と8022がAND受信するような、複雑なタスク間インタフェイスを示している。

【００５３】送信タスク8011は、イベントフラグ8001のビット7をセット（送信）し、受信タスク8012はビット8をセット（送信）する。受信タスク8021と8022は、イベントフラグ8001のビット7とビット8に対してAND受信要求（両ビットがセットされることを確認する）を発行し、受信完了後、受信タスク8021と8022はそれぞれ、イベントフラグ8001のビット7と8をクリア（送信）する。

【００５４】図9は図8に示すタスク間インタフェイスの動作例を示すシーケンス図で、送受信の順序関係により場合分けした例を示している。

【００５５】ケースAは、二つの受信タスク8021と8022が受信要求を発行後、二つの送信タスク8011と8012が送信イベントを発行した場合を示している。この場合、イベントフラグ8001のビット7と8がともにセットされた時点で、ブロックされていた受信タスク8021と8022が一斉に実行可能になる（ブロードキャストされる）。

【００５６】ケースBは、二つの送信タスク8011と8012が送信イベントを発行後、二つの受信タスク8021と8022が受信要求を発行した場合を示している。この場合、受信タスク8021と8022はブロックされることなく、受信要求が直ちに満たされる。

【００５７】ケースCは、ケースAと同様に、二つの受信タスク8021と8022が受信要求を発行後、二つの送信タスク8011と8012が送信イベントを発行した場合を示しているが、送信タスク8011がセットしたビットが受信される前に（従ってクリアされる前に）、もう一度、そのビットをセットする場合を示している。この場合のシーケンス図は、ケースAと同様であり、結局、イベントフラグ8001に対する送信イベントは蓄積効果が無いことを示している。

【００５８】図10は図8に示すタスク間インフェイスを備えたマルチタスクソフトウェアが図9に示すシーケンス図に従って動作するために必要なイベント-応答プロトコルを示す図である。

【００５９】同図(a)は、忠実にイベント-応答プロトコルを表現したものであり、イベントと応答がそれぞれAND条件になっている。

【００６０】同図(b)は、同図(a)に示すプロトコルを二つ分割したもので、ビット7とビット8がともにセットされない限り、タスク8021と8022は受信イベントを受取らないため、このような分割が可能になる。

【００６１】同図(c)は、同図(b)に示すプロトコルにおいて、どの送信タスクが送信したかを問わないプロトコルである。イベントフラグ8001のビット7と8をセットおよびクリアするタスクが8011,8012,8021,8022であることが確実であれば、同図(c)における二つのプロトコルの両立は、同図(b)の二つのプロトコルの両立と、そして同図(a)と等価である。

【００６２】［イベント-応答表現の変換］図11は図4，図7，図10に示したイベント-応答表現をより簡単な表現形式に変換する規則を示す図である。

【００６３】同図(a)の一行目は、イベントEに対して、レスポンスR1,R2の順に応答しなければならないことを示している（以下、このような場合を「手順型応答結合」と称する）。累積効果があるイベントの場合、同図(a)の二行目および三行目に示す、二つの独立なイベント-応答プロトコルに分解することができる。累積効果がないイベントの場合は、タイミングの問題が生じて、一般に、このような分解は等価ではない。

【００６４】同図(b)の一行目は、イベントEに対して、レスポンスR1とR2がすべて応答しなければならないことを示している（以下、このような場合を「AND型応答結合」と称する）。累積効果があるイベントの場合、同図(b)の二行目および三行目に示す、二つの独立なイベント-応答プロトコルに分解することができる。累積効果がないイベントの場合は、タイミングの問題が生じて、一般に、このような分解は等価ではない。

【００６５】同図(c)の一行目は、イベントE1に引続いてE2が発生した場合に、レスポンスRが応答しなければならないことを示している（以下、このような場合を「手順型イベント結合」と称する）。累積効果があるイベントの場合、同図(c)の二行目および三行目に示す、二つの独立なイベント-応答プロトコルに分解することができる。累積効果がないイベントの場合は、タイミングの問題が生じて、一般に、このような分解は等価ではない。

【００６６】同図(d)の一行目は、イベントE1またはE2

が発生した場合に、レスポンスRが応答しなければならないことを示している（以下、このような場合を「OR型イベント結合」と称する）。累積効果の有無にかかわらず、同図(d)の二行目および三行目に示す、二つの独立なイベント-応答プロトコルに分解することができる。

【００６７】上記の以外の場合（例えば、OR型応答結合やAND型イベント結合）は、簡単な分解規則は存在しないと思われる。しかし、図7(d)に示したプロトコルのように、受信タスクを特定しない方法により、OR型応答結合のプロトコルの表現を簡素化することが可能である。また、図10(b)に示したプロトコルのように応答の言い換えを行う方法により、AND型応答結合のプロトコル（イベントフラグのAND受信のときに発生する）の表現を簡素化することが可能である。

【００６８】［タスク間インタフェイス違反の検出］次に、マルチタスクモニタ1005がイベント-応答テーブルに従って、マルチタスクソフトウェアのタスク間インタフェイス違反を検出する方法について説明する。なお、以下の説明では、まず図12から図17を用いてデータ構造を説明し、次に、図18から図25を用いてそのデータ構造を用いた違反検出手続を説明する。

【００６９】［データ構造］図12はイベントおよびレスポンスを表現するイベント-応答データ構造体の一例を示す図で、タイプフィールド12001は、タスクまたはリソース、あるいはとくにリソースの場合はリソースのタイプを表現する記号常数を格納するフィールドである。IDフィールド12002は、タイプフィールド12001に従い、タスクまたはリソースのIDを格納するフィールドである。動作フィールド12003は、イベントまたはレスポンスの動作の識別子を格納するフィールドである。値フィールド12004は、イベントまたは応答の動作対象リソースに固有の値を格納するフィールドである。

【００７０】図13はイベント-応答データ構造体の詳細例を示す図で、イベントおよびレスポンスの発生対象を、タスクとリソース、さらにリソースのときはそのリソースのタイプにより場合分けして示すものである。

【００７１】同図(a)は、キューを対象としたイベント-応答データ構造体の詳細で、タイプフィールドは常に記号常数「TASK」であり、IDフィールドには動作主体のタスクIDまたはタスクを特定しない場合には「0」を格納する。動作フィールドには動作識別子の「送信」「受信」または「受信要求」を格納する。値フィールドは使わない。

【００７２】同図(b)は、イベントフラグを対象としたイベント-応答データ構造体の詳細で、タイプフィールドとIDフィールドは同図(a)と同様である。動作フィールドには動作識別子の「セット」「クリア」「送信」「受信」または「受信要求」を格納する。値フィールドには、動作フィールドの識別子が「セット」または「クリア」の場合にはイベントフラグとビット演算するビットパターンを格納し、他の識別子の場合にはイベントフラグのどのビット位置を対象とするかを示すビットマスクを格納する。

【００７３】同図(c)は、セマフォ(semaphore)を対象としたイベント-応答データ構造体の詳細で、タイプフィールドとIDフィールドは同図(a)と同様である。動作フィールドには動作識別子の「送信」「受信」または「受信要求」を格納する。値フィールドには送信または受信するセマフォ資源数を格納する。

【００７４】同図(d)は、タスクを対象としたイベント-応答データ構造体の詳細で、タイプフィールドにはそのタスクの操作対象になるリソースの種類を示す記号常数「QUEUE」「FLAG」「SEMAPHORE」を格納し、それぞれキュー，イベントフラグ，セマフォを識別する。IDフィールドにはタイプフィールドで特定されるリソースのIDを格納する。動作フィールドおよび値フィールドは、タイプが「QUEUE」の場合は同図(a)と、「FLAG」の場合は同図(b)と、「SEMAPHORE」の場合は同図(c)と、それぞれ同様である。

【００７５】図14はイベント-応答テーブル1002内の要素であるイベント-応答プロトコル構造体の一例を示す図で、モードフィールド14001は、イベント発生の累積効果があるか否かを示すフィールドである。なお、以下では、累積効果がある場合を「係数モード」、無い場合を「バイナリモード」と称する。

【００７６】励起数フィールド14002は、イベント発生後、まだレスポンスが実行されていないイベント発生数をカウントするフィールドである。モードフィールドが「係数モード」のとき、励起数フィールドは零以上の整数になり、「バイナリモード」のときは0か1になる。

【００７７】イベントフィールド14003および応答フィールド14004にはそれぞれ、イベント-応答データ構造体が格納される。手順/AND/OR結合された複合のイベントおよびレスポンスは考えない。図11で説明した方法に従い、単純なイベント-応答ペアとして、タスク間インタフェイスの仕様を表現する必要がある。

【００７８】許容時間フィールド14005は、イベント発生後、応答が実行されるまでの許容時間をシステムクロックを単位に表現した値を格納する。この許容時間を経過してもレスポンスが実行されない場合は、このイベント-応答プロトコルに違反したと解釈する。

【００７９】リンクフィールド14006は、イベント-応答プロトコル構造体をリスト状に連結するためのフィールドである。

【００８０】図15はイベントに対するレスポンスの実行を検出するために用いる励起応答構造体の一例を示す図で、イベント-応答構造体リンクフィールド15001は、イベント発生後、レスポンスが未実行のイベント-応答構造体を指すフィールドである。

【００８１】残時間フィールド15002は応答が実行され

なければならない残り時間を示すフィールドであり、前方リンクフィールド15003および後方リンクフィールド15004は励起応答構造体を双方向にリスト状に連結するために使用するフィールドである。

【００８２】図16Aおよび図16Bはタスク間インタフェイスの検査のために変更されるマルチタスクモニタ1005内のデータ構造体の一例を示す図で、図16Aに示す拡張タスクコントロールブロック（拡張TBC）16001は、タスクを管理するTCB1006をタスク間インタフェイスの検査用に拡張したものである。拡張部分のイベント-応答リスト16012には、TBC1006が管理するタスクに対するイベント-応答プロトコルのリストを指すポインタを格納する。同じく拡張部分の励起応答リスト16013には、対応する励起応答構造体のリストを指すポインタを格納する。

【００８３】また、図16Bに示す拡張リソースコントロールブロック16002は、リソースを管理するリソースコントロールブロックを拡張したものであり、拡張TBC16001と同様のイベント-応答リスト16022および励起応答リスト16023の拡張部分を備えている。

【００８４】図17は図14～図16Bに示したデータ構造体の動的なリンク構造の一例を示す図で、17001は拡張TBCまたはリソースコントロールブロック、17002はコントロールブロック17001により管理されるタスクまたはリソースに対するイベント-応答プロトコルリスト、17003は応答実行がチェックされる励起応答構造体のリストである。

【００８５】コントロールブロック17001のイベント-応答プロトコルのリストフィールドには、イベント-応答プロトコルリスト17002の先頭要素を指すポインタが格納され、励起応答リストフィールドには、励起応答構造体のリスト17003の先頭要素を指すポインタが格納される。

【００８６】［違反検出手続］図18はマルチタスクソフトウェアからマルチタスクモニタ1005にサービス要求を発行した場合にマルチタスクモニタ1005が実行する処理の一例を示すフローチャートで、ステップS181,S186はマルチタスクモニタ1005の通常の処理であり、ステップS182,S183,S185がイベント-応答プロトコルをチェックするために追加した処理である。

【００８７】ステップS182は、実行中のタスクに対して、イベント-応答リストを調べる手続の呼出しである。なお、どのタスクが実行中であるかは、マルチタスクモニタ1005により管理されている。

【００８８】ステップS183は、サービス対象のリソースに対して、イベント-応答リストを調べる手続の呼出しである。なお、サービス対象のリソースIDは、サービス要求呼出しパラメータに含まれている。

【００８９】ステップS184は実行中のタスクに対して励起応答リストを調べる手続の呼出し、ステップS185はサービス対象のリソースに対して励起応答リストを調べる手続の呼出しである。

【００９０】図19は基本クロックを通知するためにマルチタスクソフトウェアからマルチタスクモニタ1005のタイマサービスが呼出された場合にマルチタスクモニタ1005が実行する処理の一例を示すフローチャートで、ステップS191,S193はマルチタスクモニタ1005の通常の処理であり、ステップS192がイベント-応答プロトコルをチェックするために追加した処理である。ステップS192は、全タスクおよび全リソースに対して、励起応答リストのタイムアウトを調べる手続の呼出しである。

【００９１】図20はイベント-応答リストを調べてリスト中のイベント発生を検出する手続の一例を示すフローチャートで、図18に示したステップS182およびS183から呼出される手続である。同図において、ステップS201で、パラメータで指定されたイベント-応答リストの先頭要素（イベント-応答プロトコル構造体）を取出す。次に、ステップS202とS204によりリストの要素を次々に取出し、ステップS203で取出したイベント-応答プロトコル構造体に対してイベントの発生を調べる手続を呼出す。

【００９２】図21は励起応答リストを調べてリスト中の励起応答の実行を検出する手続の一例を示すフローチャートで、図18に示したステップS184およびS185から呼出される手続である。同図において、ステップS211で、パラメータで指定された励起応答リストの先頭要素（励起応答構造体）を取出す。次に、ステップS212とS214によりリストの要素を次々に取出し、ステップS213で取出した励起応答構造体に対して励起応答の実行を調べる手続を呼出す。

【００９３】図22は励起応答リスト中の励起応答のタイムアウトを検出する手続の一例を示すフローチャートで、図19に示したステップS192から呼出される手続である。同図において、ステップS221で、パラメータで指定された励起応答リストの先頭要素（励起応答構造体）を取出す。次に、ステップS222とS224によりリストの要素を次々に取出し、ステップS223で取出した励起応答構造体に対して励起応答のタイムアウトを調べる手続を呼出す。

【００９４】図23はパラメータで指定されたタスクまたはリソースに対して、さらにパラメータで指定されたイベント-応答プロトコル構造体に対するイベントの発生を検出する手続の一例を示すフローチャートで、図20に示したステップS203から呼出される手続である。同図において、ステップS231およびステップS232で、イベント-応答プロトコル構造体のモードフィールドおよび励起数フィールドを調べ、「バイナリモード」かつ励起数フィールドが正の場合は直ちにリターンする。

【００９５】次に、ステップS233とS234で、イベントフィールド（イベント-応答データ構造体）のタイプ，I

D，動作，値フィールドを比較して、もし一致しなければ、つまりイベントが発生したのでなければ直ちにリターンする。

【００９６】次に、ステップS235で、励起応答構造体のためのメモリ領域を割当て、残時間フィールドにイベント-応答プロトコル構造体の許容時間フィールドの値をコピーし、イベント-応答プロトコルリンクフィールドにイベント-応答プロトコル構造体のアドレスをセットし、パラメータで指定されたタスクまたはリソースの励起応答リストに追加する、つまり前方リンクまたは後方リンクフィールドを適切に設定する。

【００９７】次に、ステップS236で、イベント-応答プロトコル構造体の励起数フィールドをインクリメントする。

【００９８】図24はパラメータで指定された励起応答構造体に対して励起応答の実行を検出する処理の一例を示すフローチャートで、図21に示したステップS213から呼出される手続である。同図において、ステップS241で、パラメータで指定された励起応答構造体のイベント-応答プロトコルリンクフィールドの値から、該当するイベント-応答プロトコル構造体を獲得する。

【００９９】次に、ステップS242とS243で、イベント-応答プロトコル構造体の応答フィールド（イベント-応答データ構造体）のタイプ，ID，動作，値フィールドを比較して、もし一致しなければ、つまり応答が検出されなかった場合は直ちにリターンする。

【０１００】次に、ステップS244でイベント-応答プロトコル構造体の励起数フィールドをデクリメントし、ステップS245でその励起応答構造体を励起応答リストから取除き、その構造体に割当てたメモリ領域を解放する。

【０１０１】図25はパラメータで指定された励起応答構造体のタイムアウトを検出する手続の一例を示すフローチャートで、図22に示すステップS223から呼出される手続である。同図において、ステップS251とS252で、パラメータで指定された励起応答構造体の残時間フィールドをデクリメントし、その結果の残時間フィールドが正であれば直ちにリターンする。もし、残時間フィールドが零になった場合はタイムアウトが発生したことになり、この場合は、ステップS253で図1に示した通信装置1008を用いてエラーを通知し、ステップS254で励起応答構造体のイベント-応答プロトコルリンクフィールドから、対応するイベント-応答プロトコル構造体を獲得し、ステップS255でイベント-応答プロトコル構造体の励起数フィールドをデクリメントし、ステップS256で励起応答構造体を励起応答リストから取除き、励起応答構造体が占めていたメモリ領域を解放する。

【０１０２】以上説明したように、本実施形態によれば、マルチタスクソフトウェアの動作を詳細に調べることなく、タスク間インタフェイスのエラーを自動的に検出することができる。従って、タスク間インタフェイス仕様の検査により、ソフトウェアの不具合現象の原因がどのタスクにあるかが直ちに判明するため、現象の発現から対策までの時間ラグを短縮することができる。

【０１０３】さらに、本実施形態のソフトウェア検査方法は、簡便な通信装置しか備えないハードウェアのマルチタスクソフトウェアを開発する際にも、直ちに利用することができる。

【０１０４】

【他の実施形態】なお、本発明は、複数の機器（例えばホストコンピュータ，インタフェイス機器，リーダ，プリンタなど）から構成されるシステムに適用しても、一つの機器からなる装置（例えば、複写機，ファクシミリ装置など）に適用してもよい。

【０１０５】また、本発明の目的は、前述した実施形態の機能を実現するソフトウェアのプログラムコードを記録した記憶媒体を、システムあるいは装置に供給し、そのシステムあるいは装置のコンピュータ（またはCPUやMPU）が記憶媒体に格納されたプログラムコードを読出し実行することによっても、達成されることは言うまでもない。この場合、記憶媒体から読出されたプログラムコード自体が前述した実施形態の機能を実現することになり、そのプログラムコードを記憶した記憶媒体は本発明を構成することになる。プログラムコードを供給するための記憶媒体としては、例えば、フロッピディスク，ハードディスク，光ディスク，光磁気ディスク，CD-ROM，CD-R，磁気テープ，不揮発性のメモリカード，ROMなどを用いることができる。

【０１０６】また、コンピュータが読出したプログラムコードを実行することにより、前述した実施形態の機能が実現されるだけでなく、そのプログラムコードの指示に基づき、コンピュータ上で稼働しているOSなどが実際の処理の一部または全部を行い、その処理によって前述した実施形態の機能が実現される場合も含まれることは言うまでもない。

【０１０７】さらに、記憶媒体から読出されたプログラムコードが、コンピュータに挿入された機能拡張ボードやコンピュータに接続された機能拡張ユニットに備わるメモリに書込まれた後、そのプログラムコードの指示に基づき、その機能拡張ボードや機能拡張ユニットに備わるCPUなどが実際の処理の一部または全部を行い、その処理によって前述した実施形態の機能が実現される場合も含まれることは言うまでもない。

【０１０８】

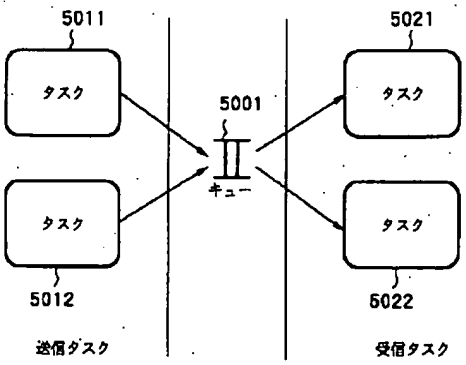
【発明の効果】以上説明したように、本発明によれば、マルチタスクソフトウェアの動作を詳細に調べることなく、タスク間インタフェイスのエラーを検出するソフトウェア検査方法およびその装置を提供することができる。

【図面の簡単な説明】

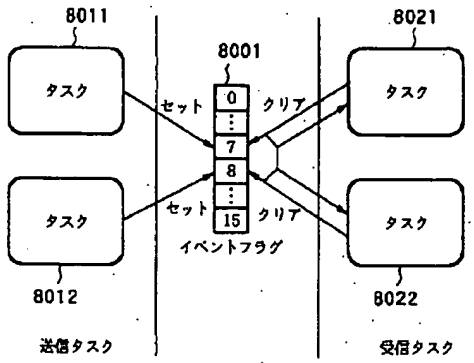
【図１】本発明にかかる一実施形態のソフトウェア検査

装置の構成例を示す機能ブロック図、
【図２】典型的なマルチタスクソフトウェアを示す図、
【図３】図2に示すマルチタスクソフトウェアの時系列的な動作例を示すシーケンス図、
【図４】タスク間インタフェイスプロトコルをイベント-応答表現で表す図、
【図５】複雑なタスク間インタフェイスの一例を示す図、
【図６】図5に示すタスク間インタフェイスの動作例を示すシーケンス図、
【図７】図5に示すタスク間インフェイスを備えたマルチタスクソフトウェアが図6に示すシーケンス図に従って動作するために必要なイベント-応答プロトコルを示す図、
【図８】複雑なタスク間インタフェイスの第二例を示す図、
【図９】図8に示すタスク間インタフェイスの動作例を示すシーケンス図、
【図１０】図8に示すタスク間インフェイスを備えたマルチタスクソフトウェアが図9に示すシーケンス図に従って動作するために必要なイベント-応答プロトコルを示す図、
【図１１】図4，図7，図10に示したイベント-応答表現をより簡単な表現形式に変換する規則を示す図、
【図１２】イベントおよびレスポンスを表現するイベント-応答データ構造体の一例を示す図、
【図１３】イベント-応答データ構造体の詳細例を示す図、
【図１４】イベント-応答テーブル内の要素であるイベント-応答プロトコル構造体の一例を示す図、
【図１５】イベントに対するレスポンスの実行を検出するために用いる励起応答構造体の一例を示す図、
【図１６Ａ】タスク間インタフェイスの検査のために変更されるマルチタスクモニタ内のデータ構造体の一例を示す図、
【図１６Ｂ】タスク間インタフェイスの検査のために変更されるマルチタスクモニタ内のデータ構造体の一例を示す図、
【図１７】図14～図16Bに示したデータ構造体の動的なリンク構造の一例を示す図、
【図１８】マルチタスクソフトウェアからマルチタスクモニタにサービス要求を発行した場合にマルチタスクモニタが実行する処理の一例を示すフローチャート、
【図１９】基本クロックを通知するためにマルチタスクソフトウェアからマルチタスクモニタのタイマサービスが呼出された場合にマルチタスクモニタが実行する処理の一例を示すフローチャート、
【図２０】イベント-応答リストを調べてリスト中のイベント発生を検出する手続の一例を示すフローチャート、
【図２１】励起応答リストを調べてリスト中の励起応答の実行を検出する手続の一例を示すフローチャート、
【図２２】励起応答リスト中の励起応答のタイムアウトを検出する手続の一例を示すフローチャート、
【図２３】パラメータで指定されたタスクまたはリソースに対して、さらにパラメータで指定されたイベント-応答プロトコル構造体に対するイベントの発生を検出する手続の一例を示すフローチャート、
【図２４】パラメータで指定された励起応答構造体に対して励起応答の実行を検出する処理の一例を示すフローチャート、
【図２５】パラメータで指定された励起応答構造体のタイムアウトを検出する手続の一例を示すフローチャートである。

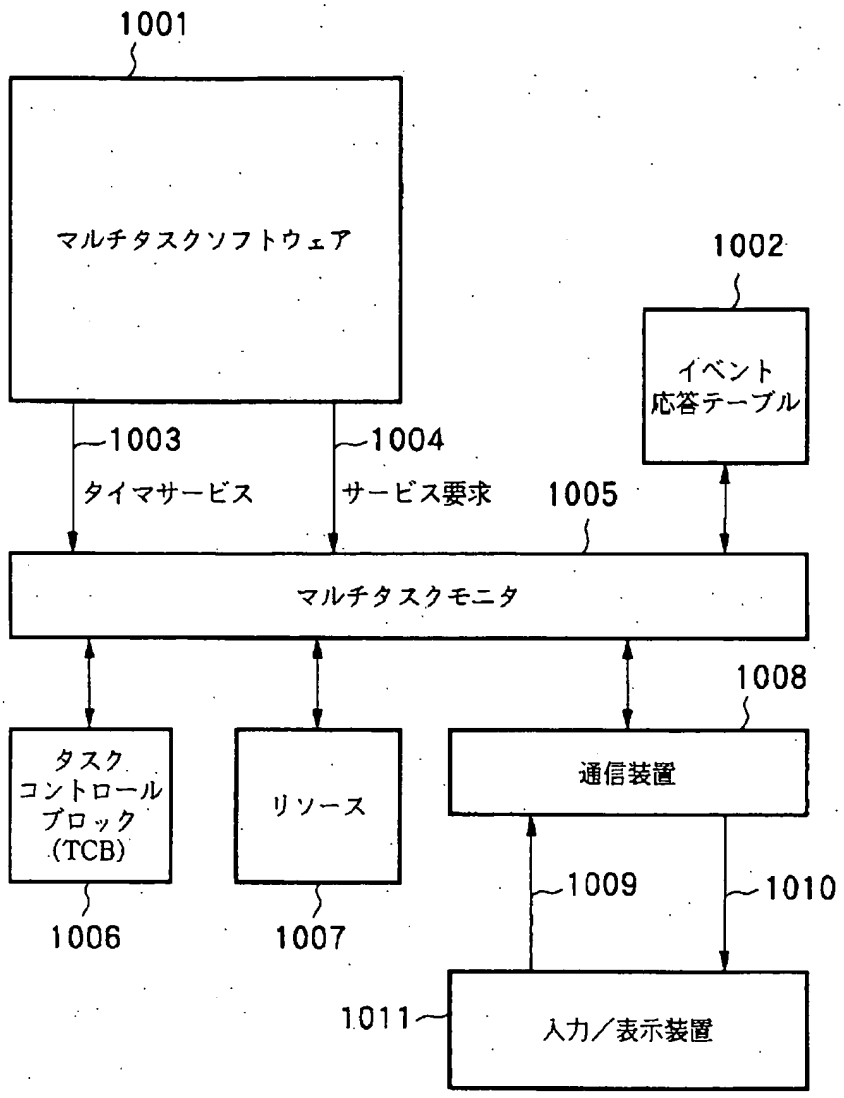
【図５】



【図８】



【図1】



【図4】

キュー2021	
イベント	応答
2051送信	2052受信

キュー2022	
イベント	応答
2055送信	2056受信

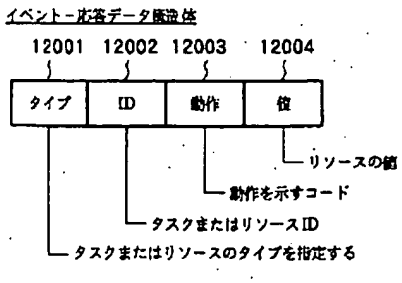
イベントフラグ2031	
イベント	応答
2053送信	2054受信
2057送信	2058受信

タスク2011	
イベント	応答
2052受信	2053送信、2058要求
2058受信	2052要求

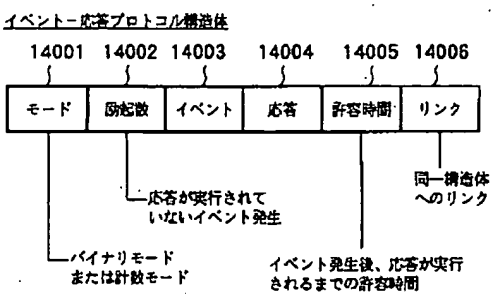
タスク2012	
イベント	応答
2054受信	2055送信、2054要求

タスク2013	
イベント	応答
2056受信	2057送信、2066要求

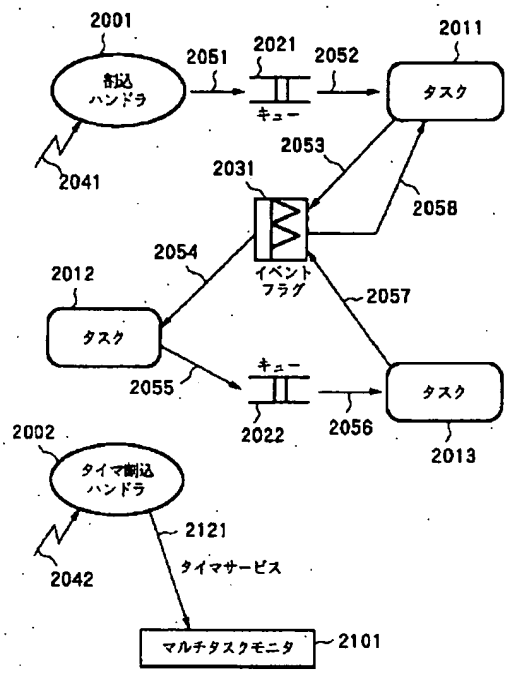
【図12】



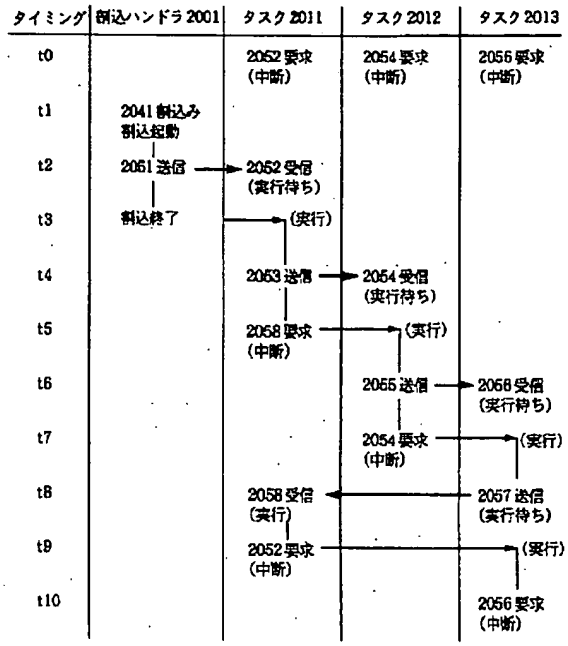
【図14】



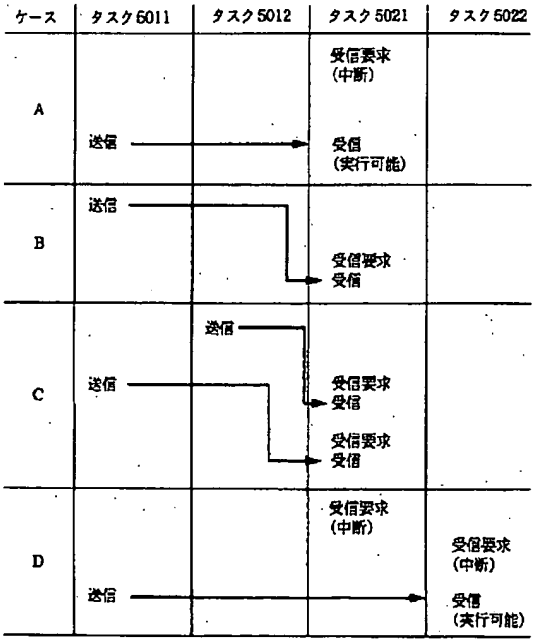
【図2】



【図3】



【図6】



【図7】

(a)

イベント－応答プロトコル		
イベント	応答	累積効果
タスク 5011 または 5012 が送信した	タスク 5021 または 5022 が受信する	有り

(b)

分割されたイベント－応答プロトコル		
イベント	応答	累積効果
タスク 5011 が送信した	タスク 5021 または 5022 が受信する	有り
タスク 5012 が送信した	タスク 5021 または 5022 が受信する	有り

(c)

送信者を省略したイベント－応答プロトコル		
イベント	応答	累積効果
タスクの何れかが送信した	タスク 5021 または 5022 が受信する	有り

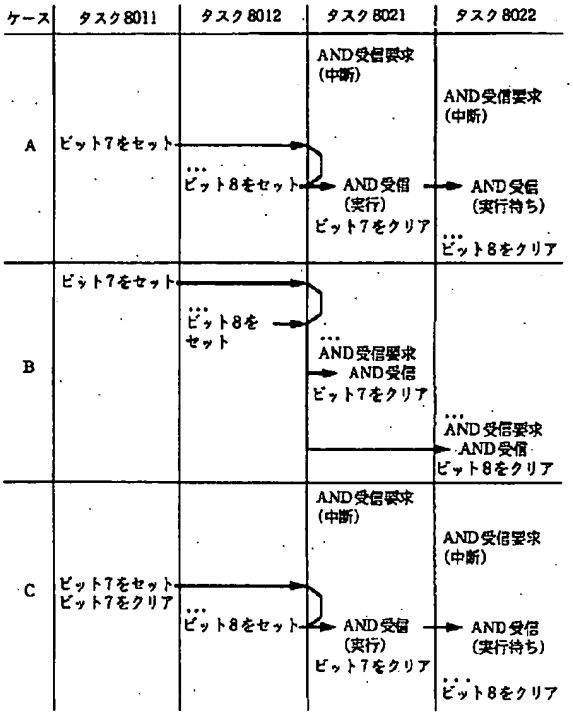
(d)

受信者を省略したイベント－応答プロトコル		
イベント	応答	累積効果
タスク 5011 が送信した	タスクの何れかが受信する	有り
タスク 5012 が送信した	タスクの何れかが受信する	有り

(e)

送信者，受信者を省略したイベント－応答プロトコル		
イベント	応答	累積効果
タスクの何れかが送信した	タスクの何れかが受信する	有り

【図9】



【図10】

(a)

イベント－応答プロトコル		
イベント	応答	累積効果
タスク8011がビット7をセットし、タスク8012がビット8をセットした	タスク8021と8022がAND受信する	無し

(b)

分割したイベント－応答プロトコル		
イベント	応答	累積効果
タスク8011がビット7をセットした	タスク8021がAND受信する	無し
タスク8012がビット8をセットした	タスク8022がAND受信する	無し

(c)

送信者を省略したイベント－応答プロトコル		
イベント	応答	累積効果
ビット7がセットされた	タスク8021がAND受信する	無し
ビット8がセットされた	タスク8022がAND受信する	無し

【図11】

(a)

手順型応答結合	
イベント	レスポンス
E	R1, 続いてR2
E	R1
R1	R2

(b)

AND型応答結合	
イベント	レスポンス
E	R1, そしてR2
E	R1
E	R2

(c)

手順型イベント結合	
イベント	レスポンス
E1, 続いてE2	R
E1	E2
E2	R

(d)

OR型応答結合	
イベント	レスポンス
E1, またはE2	R
E1	R
E2	R

【図13】

(a)

キューに対するイベント－応答データ構造体			
タイプ	ID	動作	値
TASK	タスクID または0	送信	—
		受信要求	
		受信	

(b)

イベントフラグに対するイベント－応答データ構造体			
タイプ	ID	動作	値
TASK	タスクID または0	セット	ビットパターン
		クリア	
		AND受信要求	ビットマスク
		AND受信	
		OR受信要求	
		OR受信	

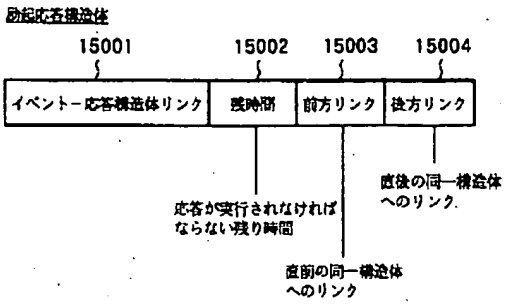
(c)

セマフォに対するイベント－応答データ構造体			
タイプ	ID	動作	値
TASK	タスクID または0	送信	資源数
		受信要求	
		受信	

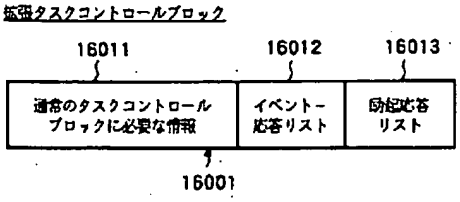
(d)

タスクに対するイベント－応答データ構造体		
タイプ	ID	動作および値
QUEUE	キューID	(a) に同じ
FLAG	イベントフラグID	(b) に同じ
SEMAPHORE	セマフォID	(c) に同じ

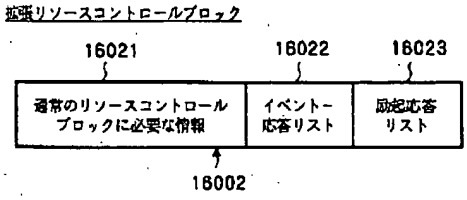
【図15】



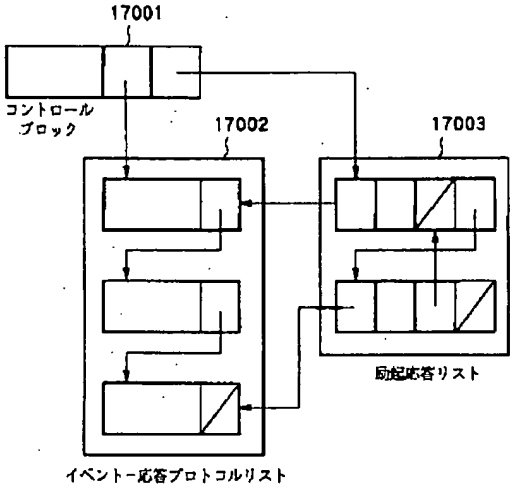
【図16A】



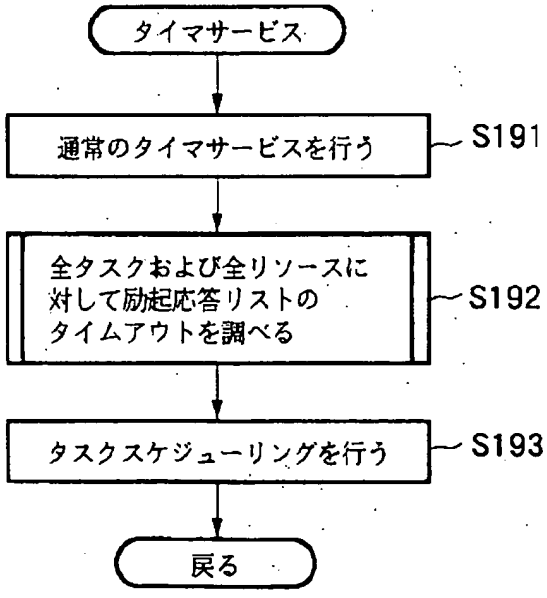
【図16B】



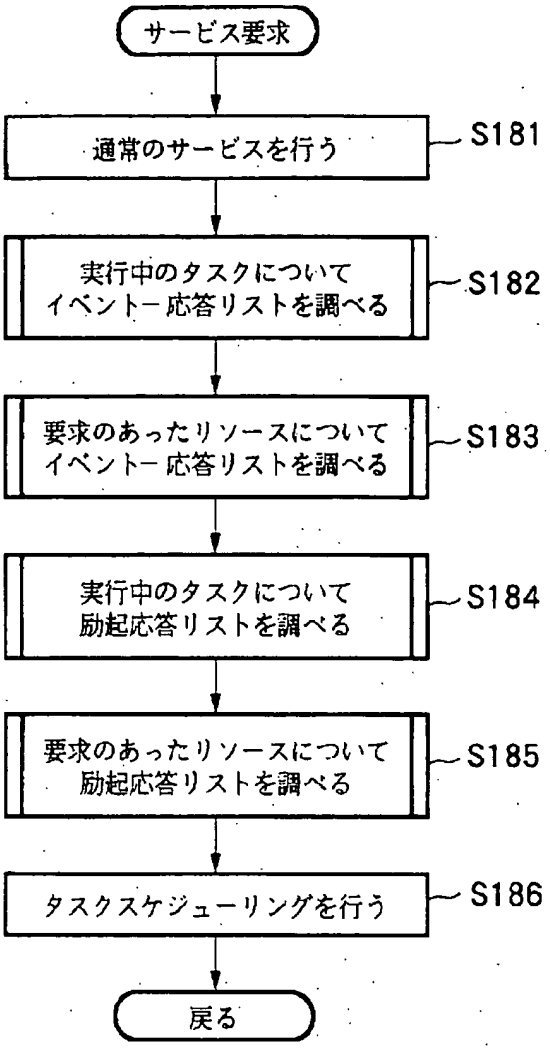
【図17】



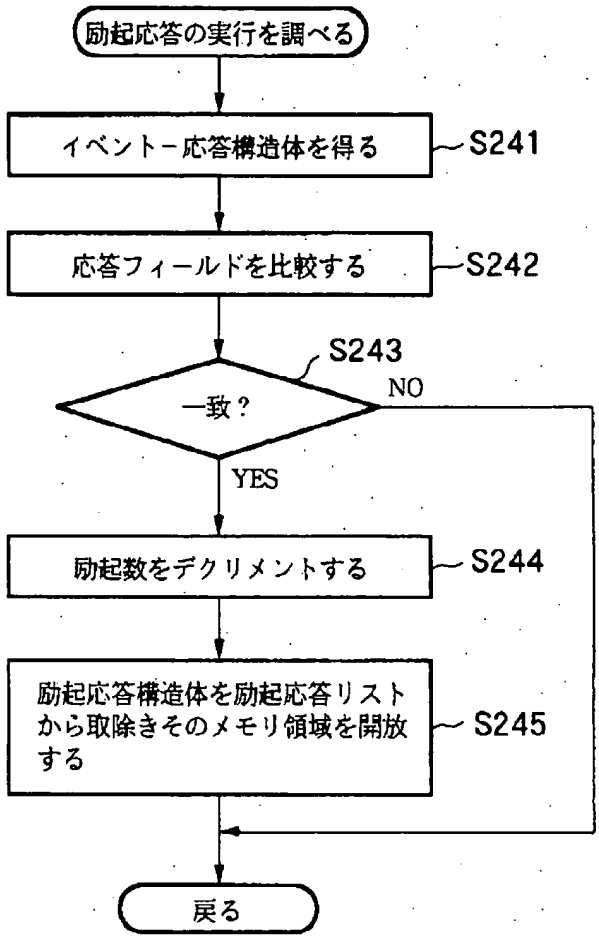
【図19】



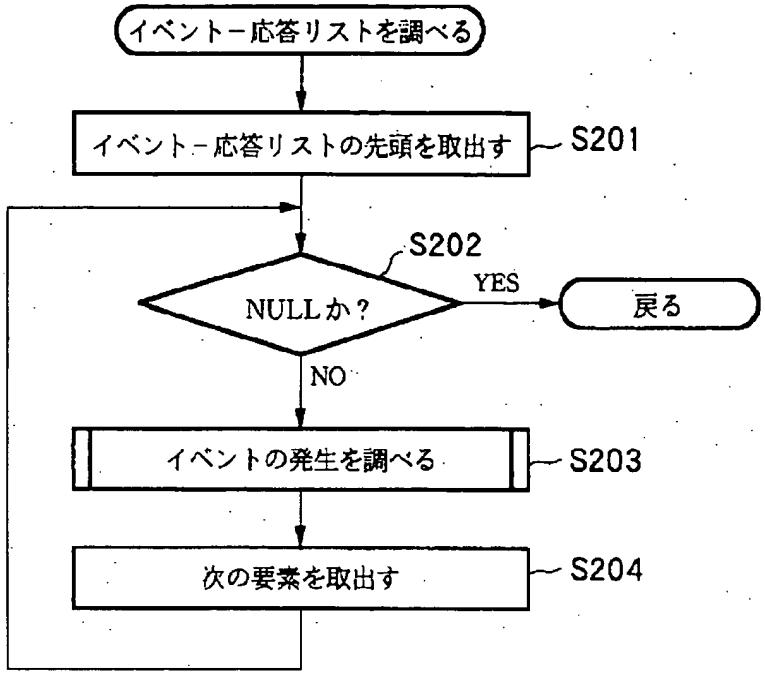
【図18】


サービス要求
通常のサービスを行う
S181
実行中のタスクについて
イベント－応答リストを調べる
S182
要求のあったリソースについて
イベント－応答リストを調べる
S183
実行中のタスクについて
励起応答リストを調べる
S184
要求のあったリソースについて
励起応答リストを調べる
S185
タスクスケジューリングを行う
S186
戻る

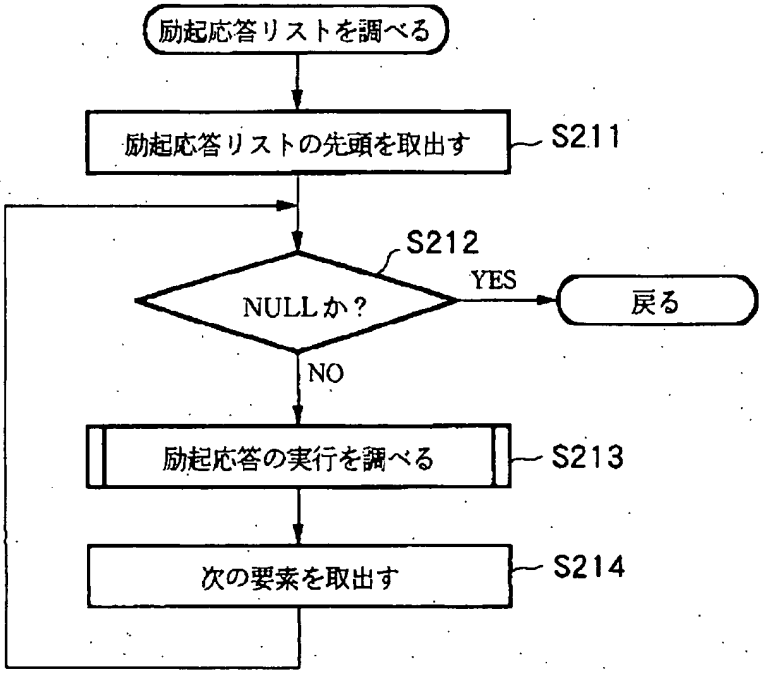
【図24】


励起応答の実行を調べる
イベント－応答構造体を得る
S241
応答フィールドを比較する
S242
S243
一致？
NO
YES
励起数をデクリメントする
S244
励起応答構造体を励起応答リスト
から取除きそのメモリ領域を開放
する
S245
戻る

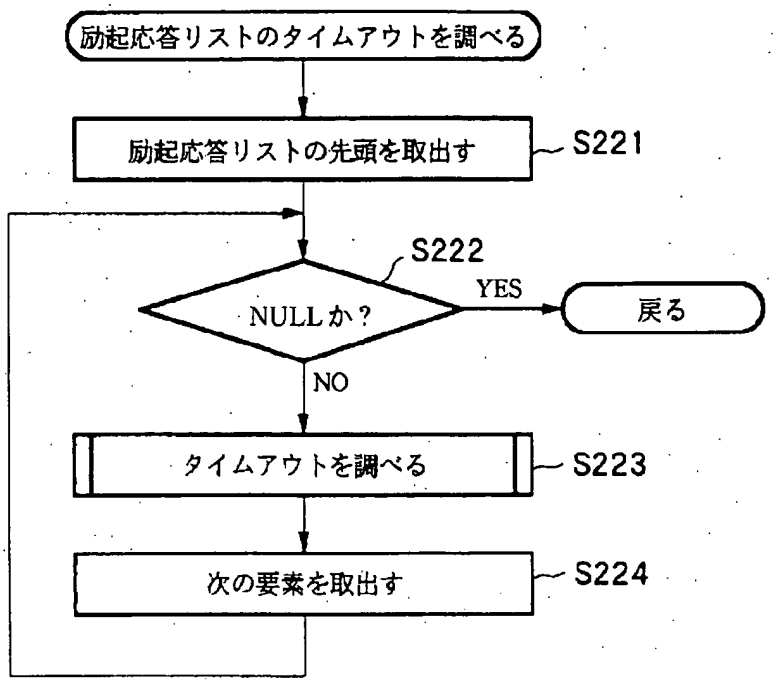
【図20】


イベント－応答リストを調べる
イベント－応答リストの先頭を取出す
S201
S202
NULLか？
YES
戻る
NO
イベントの発生を調べる
S203
次の要素を取出す
S204

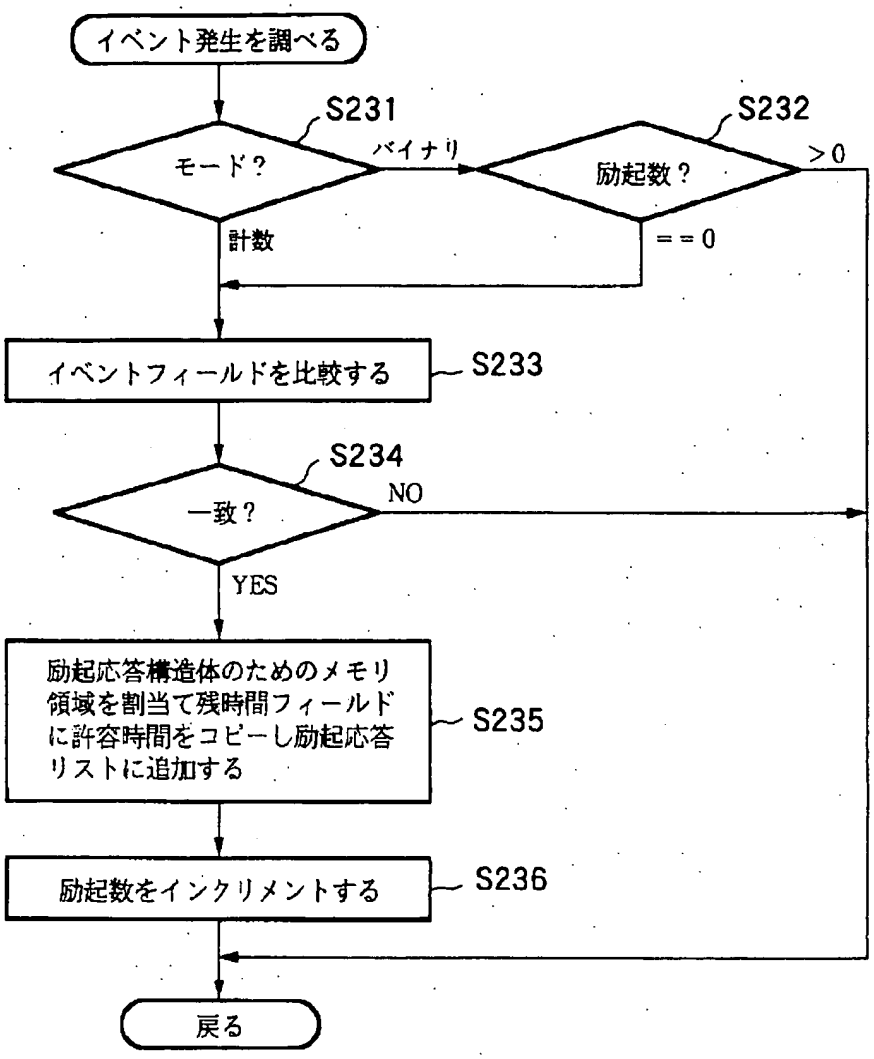
【図21】


励起応答リストを調べる
励起応答リストの先頭を取出す
S211
S212
NULLか？
YES
戻る
NO
励起応答の実行を調べる
S213
次の要素を取出す
S214

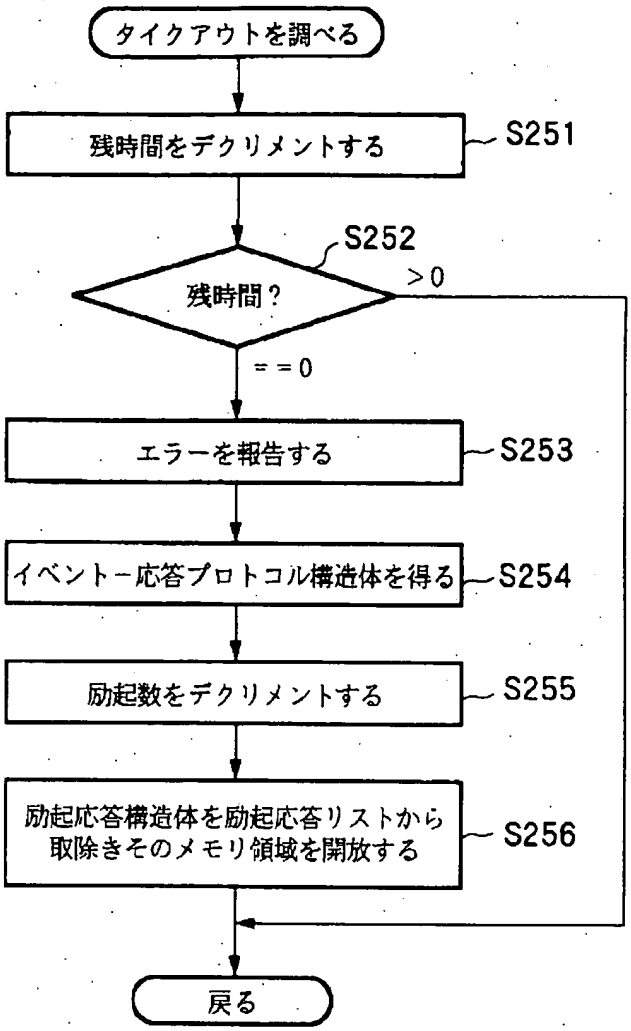
【図22】


励起応答リストのタイムアウトを調べる
励起応答リストの先頭を取出す
S221
S222
NULLか？
YES
戻る
NO
タイムアウトを調べる
S223
次の要素を取出す
S224

【図23】


イベント発生を調べる
S231
モード？
バイナリ
S232
励起数？
>0
計数
==0
イベントフィールドを比較する
S233
S234
一致？
NO
YES
励起応答構造体のためのメモリ領域を割当て残時間フィールドに許容時間をコピーし励起応答リストに追加する
S235
励起数をインクリメントする
S236
戻る

【図25】


タイクアウトを調べる
残時間をデクリメントする
S251
S252
残時間？
＞0
＝＝0
エラーを報告する
S253
イベント－応答プロトコル構造体を得る
S254
励起数をデクリメントする
S255
励起応答構造体を励起応答リストから
取除きそのメモリ領域を開放する
S256
戻る

Requested Patent: JP9185528A

Title: METHOD AND DEVICE FOR INSPECTING SOFTWARE ;

Abstracted Patent: JP9185528 ;

Publication Date: 1997-07-15 ;

Inventor(s): SASAKI SEIJI; HIRAHARA ATSUSHI; WATANABE HIROYASU ;

Applicant(s): CANON INC ;

Application Number: JP19960000462 19960108 ;

Priority Number(s): ;

IPC Classification: G06F11/28; G06F9/46 ;

Equivalents: ;

ABSTRACT:

PROBLEM TO BE SOLVED: To minutely inspect the operation of multi-task software for detecting the error of inter-task interface.SOLUTION: Multi-task software 1001 operating on hardware loading a low speed communication equipment is inspected by operating multi-task software 1001 on a real time basis and automatically detecting violence against interface specification between tasks, which is previously set, and incorporating an inspection method for inspecting multi-task software 1001 into a multi-task monitor 1005.